Bericht Todesstern U2

Charline Waldrich, Robert Ullmann, Julian Dobrot

13. november 2015

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Strahl

Die Klasse Ray representiert einen Strahl. Im konstruktor dieser Klasse soll der Ursprung und die Richtung des Strahls übergeben werden.

1.2 Kameras

Es sollen drei Kameraklassen implementiert werden.

1.2.1 Camera

Diese Klasse ist die abstrakte Basisklasse. sie speichert die drei Vektoren e,g,t welche im Konstruktor übergeben werden. Außerdem wird die Kameratransformation der vektoren u,v,w auch in ihrem Konstruktor berechnet.

1.2.2 OrthographicCamera

Soll eine orthogrsphische Kamera wiederspiegeln und hat als weiteren Parameter den Skalierungsfaktor der bildebene.

1.2.3 PespectiveCamera

Eine perspektivische Kamera mit dem Öffnungswinkel als weiterem Parameter

1.3 Farben

Die Klasse Farben soll Farben im RGB-Farbraum representieren. Dabei sollen die einzelnen Komponenten einen Wert von 0 und 1 annehmen. Farben sollen addiert, subtrahiert und multipliziert werden. Desweiteren sollen Farben mit einem skalar mutlipliziert werden können.

1.4 Geometrien

Alle Geometrien erben von einer abstrakten Basisklasse. Diese hat nur ein Attribut nämlich die Farbe.

1.4.1 Hit

1.4.2 Plane

Die Klasse Plane soll eine Implementierung einer Ebenel darstellen.

1.4.3 Sphere

Die Klasse Sphere soll eine Implementierung einer Kugel darstellen.

1.4.4 Triangle

Die Klasse Triangle soll eine Implementierung eines Dreicks darstellen.

1.4.5 AxisAlignedBox

Die Klasse Sphere soll eine Implementierung einer Box darstellen.

1.5 Welt

In der Welt sollen die darzustellenden Objekte in einer Scene dargestellt werden

1.6 Raytracer

Hier soll über jeden pixel des Bildes iteriert werden. Werden Schnittpunkte mit Geometrien gefunden, erhält der Pixel dessen Farbe.

2 Lösungsstrategien

Allgemein wurden als erster Lösungsschritt alle aus der Aufgabenstllung hervorgehenden Klassendiagramme komplett in das Projekt integriert.

2.1 Strahl

2.2 Kameras

2.2.1 Camera

2.2.2 OrthographicCamera

2.2.3 PespectiveCamera

2.3 Farben

2.4 Geometrien

2.4.1 Hit

2.4.2 Plane

2.4.3 Sphere

Als Lösungsansatz für diese Geometrie dienten die aus der Vorlesung erarbeite Formel zur berechnung der Schnittpunkte verwendet.

$t = \frac{-b \pm \sqrt[2]{b^2 - 4ac}}{2a}$

Für die berechnung der Anzahl der Schnittpunkte mit der Sphäre wurde die folgende Formel herangezohgen:

$d = b^2 - 4ac$

Wobei der Strahl bei $d < 0$ keinen Schnittpunkt mit der Kugel aufweist. Bei $d = 0$ genau einen und bei $d > 0$ zwei Schnittpunkte.
Die einzelnen Komponennten dieser formeln setzen sich wie folgt zusammen:

$a = d^2$

$b = d * [2(o - c)]$

$c = (o - c) * (o - c) - r^2$

Diese Überlegungen wurden dementsprechend programmtechnisch umgesetzt, um eine gewünschte implementierung der hit-Methode zu erzielen.

2.4.4 Triangle

2.4.5 AxisAlignedBox

2.5 Welt

2.6 Raytracer

3 Implementierungen

3.1 Strahl

Die ,,Ray" Klasse initialisiert einen Strahl mit gegebenem Ausgangspunkt und Richtungsvektor. Sie implementiert zwei Methoden namens ,,at" und ,,tOf". Außerdem wurden die von der Oberklasse Object geerbten Methoden hashCode und eqals so überschrieben, dass sie zum Testen nützlich sind.
Der Strahl (Ray) kann mit der Methode ,,at" den Punkt berechnen, welchen er trifft wenn er die als Parameter übergebenen Länge ,,t" (als double Wert), den Richtungsvektor "entlang geht". Hierfür wird

die eine einfache Rechnung angewandt.

$$PuntP = AusgangspunktO + Länget \times Richtungsvektord$$

In der Methode ,,tOf" wird diese Rechnung umgedreht. Ein gesuchter Punkt wird als Parameter übergeben und berechnet dann den Faktor ,,t" (als double Wert) welcher mit dem Richtungsvektor multipliziert werden müsste um zu gesuchtem Punkt zu gelangen.

3.2 Kameras

3.2.1 Camera

3.2.2 OrthographicCamera

3.2.3 PespectiveCamera

3.3 Farben

3.4 Geometrien

3.4.1 Hit

Das Hit Objekt bekommt einen double Wert ,,t", einen Strahl ,,ray" und eine geometrische Figur übergeben und zugewiesen. Außerdem werden in der Klasse die von der Oberklasse Object vererbten Methoden ,,toString", ,,hashCode" und ,,equals" überschrieben.

3.4.2 Plane

3.4.3 Sphere

3.4.4 Triangle

3.4.5 AxisAlignedBox

3.4.6 Plane

Die Klasse Plane beschreibt eine Ebene. Diese wird genau definiert durch die übergebenen Parameter; einen auf der Ebene liegenden Punkt, einem Normalen-Vektor welcher die Ausrichtung (bzw. Neigung) der Ebene definiert, und einem Color Objekt welche die Farbe der Ebene hält.
Die von der Oberklasse Object geerbten Methoden ,,toString", ,,hashCode" und ,,equals" wurden sinnvoll überschrieben.
Des weiteren wird die Methode ,,hit" von der abstrakten Oberklasse Geometry vererbt und auf die Ebene angepasst. Diese bekommt einen Strahl übergeben welcher nicht null sein darf und gibt ein Hit Objekt zurück. Danach wird zunächst getestet ob das Skalarprodukt des Strahls und des Normalenvektor gleich Null ist. Ist dies der Fall sind Ebene und Strahl paralell und ,,null" wird zurück gegeben.
Ist dies nicht der Fall, wird für den Strahl und die Ebene die Entfernung berechnet.
Ist diese kleiner Null bedeutet das, dass das Objekt vor dem Ortsvektor des Strahls (und somit vor unserer Bildschirmfläche) liegt und es wird wieder ein ,,null" Objekt zurück geliefert.
Ist der berechnete double-Wert größer 0 wird ein Hit Objekt mit dem berechneten ,,t"-Wert, dem der Methode übergebenen Ray ,,r" und der im Konstruktor zugewiesenen Farbe zurück gegeben (siehe Abb. 1).
Außerdem wird in unserer Color Klasse der RGB-Wert berechnet, welcher für das färben des writableImage Objektes nicht brauchbar ist. Da der pixelWriter nur Color Objet aus dem javaFX Paket erkennt,

müssen die einzeln berechneten Werte an ein neues Color Objekt dessen übergeben werden.

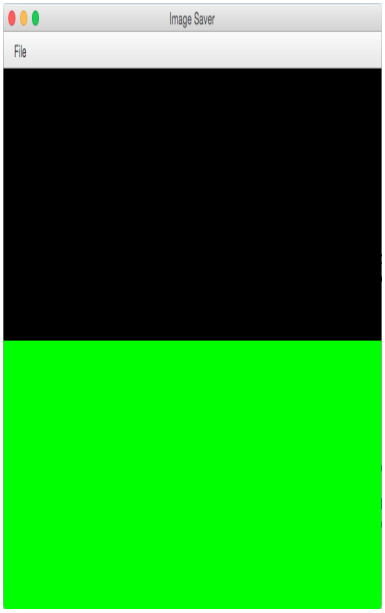


Abbildung 1: Ebene aus Perspektivischer Kamera

3.5 Welt

Die Klasse ,,World" initialisiert ein Fenster mit übergebener Hintergrundfarbe. Diese soll zukünftig vom Nutzer ausgewählt werden. Sie beinhaltet eine Array Liste in welcher nur Objekte des Typs ,,Geometry" gespeichert werden können.
Des weiteren gibt es eine Methode namens ,,hit" welche einen Strahl (Ray) übergeben bekommt und eine Farbe vom Typ Color zurück gibt. Zunächst wird ein Hit Objekt ,,hit0" der mit dem null-Wert initialisiert.
In der for-Schleife wird für jedes Objekt in der Liste ein weiteres Hit

Objekt mit dem übergebenen Strahl gespeichert. Hält das ,,hit1" Objekt einen Wert welcher ungleich null ist und liegt vor den anderen Objekten (hat also ein kleineren ,,t"-Wert, als die Objekte welche davor in hit0 gespeichert wurden), so wird der außen liegenden Variable ,,hit0" dieses Hit Objekt zugewiesen. Sollte nach dem Durchgang der kompletten Schleife der Wert von ,,hit0" immer noch null sein, so wird die Hintergrundfarbe der Welt zurück gegeben. Wenn nicht so wird immer die Farbe des am nächstliegendsten Objekt zurück gegeben.

3.6 Raytracer

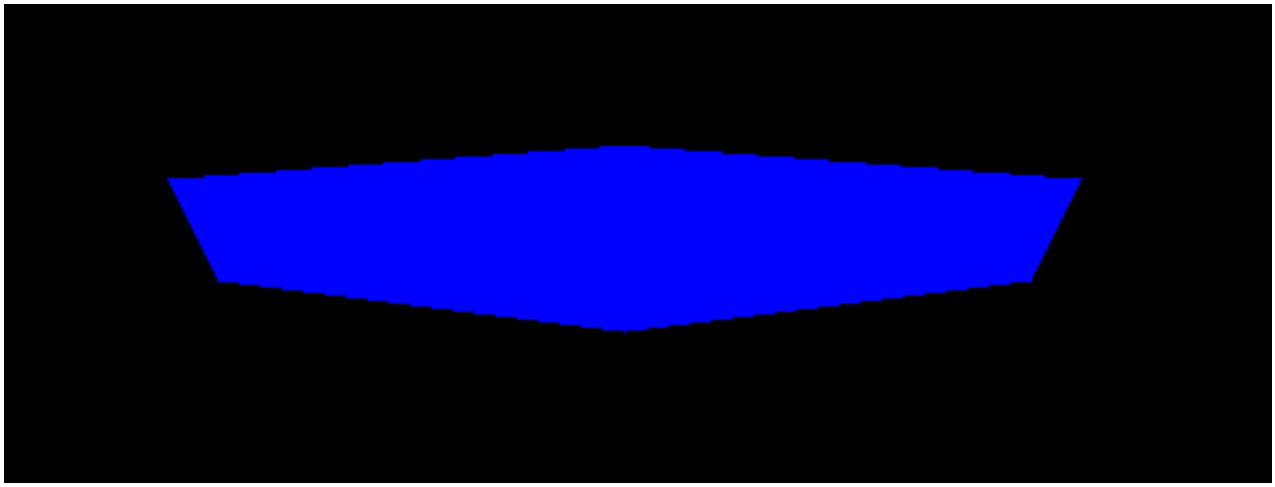
4 Besondere Probleme oder Schwierigkeiten bei der Bearbeitung

4.1 Strahl

4.2 Kameras

4.2.1 Camera

Ein Fehler in der Camera klasse führte zueiner überarbeitung der Cameratransformation, da Koordinaten falsch berechnetwurden, kam es bem Rendering der Box zu foldendem Ergebniss.



4.2.2 OrthographicCamera

4.2.3 PespectiveCamera

4.3 Farben

4.4 Geometrien

4.4.1 Hit

4.4.2 Plane

4.4.3 Sphere

Ein Fehler in einer der Methoden der Klasse Vector3 führte zu einer längeren Fehlersuche. Durch einen Operatorschreibfehler kam es zu einer Fehlerhaften berechnung wie im folgendem Bild zu sehen ist:

4.4.4 Triangle

4.4.5 AxisAlignedBox

4.5 Welt

4.6 Raytracer

In der ImageSaver Klasse musste die ,,getColor"-Methode angepasst werden. Diese benutzt unsere persönliche Color Klasse und berechnet für jeden einzelnen Pixel dank der ,,hit" Methode der Welt, ob das erzeugte Objekt von dem Strahl der erzeugten Klasse getroffen wird. Da in der Bildverarbeitung der Ursprung des Koordinatensystems oben links und in der Mathematik unten links liegt, musste bei der Berechnung aufgepasst werden, dass der ,,rayFor" Methode welche von der ,,camera" aufgerufen wird. nicht die oben liegende Y-Koordinate über-

geben wird, sondern dem quasi spiegelverkehrtem Pixel unten.

-¿ hier kannst du rein schreiben was bei der Camera Klasse das Problem war und wie es schön das ganze Projekt kaputt gemacht hat ;)

5 Zeitbedarf

Strahl	80 min
Kameras	240 min
Farben	240 min
Geometrien	240 min
Welt	240 min
Raytracer	240 min
Bericht	180 min
	740 min

5.1 Tests

6 Quellen

https://asalga.wordpress.com/2012/09/23/understanding-vector-reflection-visually/